Aperture SDK Overview

Developer

Contents

Aperture 3.4 SDK Overview 4
Plug-in Concepts 5
Versions of the Aperture SDK 5
Export Plug-in Protocols 6
Building an Export Plug-in 6
    An Example Export Plug-in 7
    The Info.plist for an Export Plug-in 7
Edit Plug-in Protocols 9
Building an Edit Plug-in 10
    Example Edit Plug-ins 10
    The Info.plist for an Edit Plug-in 10
Plug-in Locations 13

Document Revision History 15

2013-06-05 | Copyright © 2013 Apple Inc. All Rights Reserved.

Listings

Aperture 3.4 SDK Overview 4
Listing 1-1 An Export Plug-in Info.plist 7
Listing 1-2 An Edit Plug-in Info.plist 11

2013-06-05 | Copyright © 2013 Apple Inc. All Rights Reserved.

Aperture 3.4 SDK Overview

You can use the Aperture SDK to create two different kinds of Aperture plug-ins: Export plug-ins and Edit plug-ins.

Export plug-ins let you control the entire Aperture export process. With an Export plug-in, you can:

- Manage the visibility of the Aperture controls in the export window, such as Version or Master export, Export Presets, and File Naming Policy.
- Display a custom user interface within the Aperture export window.
- Provide users with custom export presets.
- Track the type of image (Master or Version) being exported.

Because of its non-destructive editing, Aperture must first generate the appropriate images before letting a plug-in perform an export operation. Users control the specific options that go into the image generation process, but an Aperture Export plug-in controls the export options that are available to the user.

You can create an Export plug-in with the Aperture 1.5.5 SDK or later.

Edit plug-ins let you modify Aperture images. With an Edit plug-in, you can:

- Request editable versions of a given image.
- Display a custom interface.
- Import result images back into the Aperture library.
- Add metadata to images.
- Manipulate image data for images in the Aperture library.

You can create an Edit plug-in with the Aperture 2.1 SDK and later.

2013-06-05 | Copyright © 2013 Apple Inc. All Rights Reserved.

Plug-in Concepts

The Aperture Export and Edit plug-ins are based on Apple's ProPlug architecture. ProPlug provides two key capabilities:

- The host application can discover installed plug-ins. By including a set of ProPlug-specific key-value pairs in a plug-ins `Info.plist` file, you provide all the data that ProPlug needs to identify your plug-in and the protocols it implements.
- The plug-in can access the host applications API objects. For example, an Aperture export plug-in can ask ProPlug for an ApertureExportManager object. (Methods for access are specified in the header file `PROAPIAccessing.h`.)

Versions of the Aperture SDK

The Aperture 3.4 SDK provides the same functionality as the Aperture 2.1 SDK, but in addition, provides templates and samples that build with Xcode 4.4 and later, and on either OS X 10.7.5 or 10.8.3 and later.

The Aperture 2.1 SDK and later provide the ability to build Edit plug-ins. These plug-ins are compatible with the Edit functionality built into Aperture 2.1 and later.

The Aperture 1.5.5 SDK and later lets you create plug-ins that run under Aperture 1.5 or later. Aperture 1.5.1 introduced several changes to the `ApertureExportManager` protocol that are not available in Aperture 1.5. In particular, version 2 of the `ApertureExportManager` protocol includes changes for hierarchical keywords, thumbnails, and image properties. Specifically:

- You can now obtain a dictionary of image properties that does not include a thumbnail by using `propertiesWithoutThumbnailForImageAtIndex:`.
- You can now obtain various sizes of a given thumbnail using the method `thumbnailForImageAtIndex:size:` and the constants `kExportThumbnailSizeThumbnail`, `kExportThumbnailSizeMini`, and `kExportThumbnailSizeTiny`.
- The dictionary returned from `propertiesForImageAtIndex:` and `propertiesWithoutThumbnailForImageAtIndex` includes an object containing the hierarchy of each keyword attached to the image. The key for that object is `kExportKeyHierarchicalKeywords`.
- You can now add keywords to an image, including the full hierarchy for each keyword using `addHierarchicalKeywords:toImageAtIndex:`.

Before using any of the updated Export functionality, your plug-in must first check whether version 2 of the `ApertureExportManager` protocol is supported by the version of Aperture the plug-in is currently running under. To do this, you can call the following method from the `PROAPIAccessing` header file:

2013-06-05 | Copyright © 2013 Apple Inc. All Rights Reserved.

```
- (BOOL)conformsToProtocol:(Protocol *)aProtocol version:(unsigned int)versionNumber;
```

For example, you can decide which SDK features to use based on the results of calling this method:

```
if ([_exportManager conformsToProtocol:@protocol(ApertureExportManager) version:2])
    // Use Export API features only available in Aperture 1.5.1 and later
else
    // Use Export API features available in Aperture 1.5 and later
```

Export Plug-in Protocols

The protocols for an Export plug-in in the Aperture SDK include:

ApertureExportPlugIn

Specified in the header file `ApertureExportPlugIn.h`. This protocol contains methods for:

- Controlling the appearance and UI of the export window.
- Controlling the export process.
- Providing progress information.

ApertureExportManager

Specified in the header file `ApertureExportManager.h`. A plug-in uses the methods in this protocol to communicate with Aperture during the export process. Version 2 of the protocol is supported by Aperture 1.5.1 and later.

PROAPIAccessing

Specified in the header file `PROAPIAccessing.h`. This protocol acts as the broker between Aperture and a plug-in. It allows a plug-in to ask which versions of a protocol the host application supports.

Building an Export Plug-in

The easiest way to build a new export plug-in for Aperture is to create a new project in Xcode using the Aperture Export Plug-In project template. (This template is included in the SDK install.) Be sure to customize the class names and fill in the proper UUIDs to avoid namespace collisions.

2013-06-05 | Copyright © 2013 Apple Inc. All Rights Reserved.

An Example Export Plug-in

An example Export plug-in, which provides simple FTP upload capability, is installed in `/Developer/Examples/Aperture/`. This example illustrates API usage as well as several more advanced concepts.

The Info.plist for an Export Plug-in

An Aperture Export plug-in is packaged in a CFBundle. Each bundle contains one or more plug-ins. A bundle also contains an `Info.plist` file.

This `Info.plist` file encodes key-value pairs that are specific to the ProPlug plug-in architecture. They tell the plug-in manager what protocols and versions the plug-in implements and what host protocols and versions it supports. As well, the key-value pairs specify appropriate names, IDs, and groupings.

Here is a commented `Info.plist` template for an Aperture Export plug-in:

Listing 1-1 An Export Plug-in Info.plist

```
<key>ProPlugDictionaryVersion</key>
<string>1.0</string>
<!--Required. Identifies the version of the ProPlug dictionary this plug-in uses.
 The value should be 1.0 for this version of the Aperture SDK.-->

<key>ProPlugDynamicRegistration</key>
<false/>
<!--Required. Tells the plug-in manager whether the plug-in principal class performs
 plug-in registration tasks. If true, the class must implement the
PROPluginRegisterBundle protocol.-->

<key>ProPlugInGroupList</key>
<!--Required, along with all child keys. Identifies this plug-in as an Export
plug-in.-->
<array>
    <dict>
        <key>groupName</key>
        <string>Export</string>
        <key>uuid</key>
        <string>616BA321-B4C2-49DF-8FD8-2E3392D2D240</string>
    </dict>
```

2013-06-05 | Copyright © 2013 Apple Inc. All Rights Reserved.

```
</array>

<key>ProPlugPlugInList</key>
<!--Required. Identifies the plug-in's in this bundle.-->
<array>
    <dict>
        <key>className</key>
        <string>xxxxxxxxxxx</string>
        <!--Required. The name of the class that implements the plug-in
        protocol.-->

        <key>displayName</key>
        <string>xxxxxxxxxxx</string>
        <!--Required. The name that will appear in menus and on the export
         window.-->

        <key>helpURL</key>
        <string>xxxxxxxxxxx</string>
        <!--Optional. If provided, Aperture displays a Help button on the
        export window. If the user clicks the Help button, Aperture launches
        the URL specified here.-->

        <key>protocolNames</key>
        <array>
            <string>ApertureExportPlugIn</string>
        </array>
        <!--Required. Specifies the plug-in protocol this plug-in implements.
        This value should be ApertureExportPlugIn for this version of the
        Aperture SDK.-->

       <key>infoString</key>
       <string>Description of your plug-in and what it does</string>
        <!-- Optional. This is the descriptive text that displays in the Aperture
        Command Customization window.-->
```

2013-06-05 | Copyright © 2013 Apple Inc. All Rights Reserved.

```
        <key>uuid</key>
        <string>xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx</string>
        <!--Required. Specifies a unique identifier for this plug-in. Use the
        uuidgen command-line tool to generate a unique identifier.-->

        <key>version</key>
        <string>1</string>
    </dict>
</array>

<key>ProPlugProtocolList</key>
<!--Required, along with all child keys. Specifies the host protocols that this
plug-in supports.-->
<array>
    <dict>
        <key>protocolName</key>
        <string>ApertureExportManager</string>
        <key>versions</key>
        <array>
            <integer>1</integer>
        </array>
    </dict>
</array>
```

Edit Plug-in Protocols

The protocols for an Edit plug-in in the Aperture 2.1 SDK include:

ApertureEditPlugIn

Specified in the header file `ApertureEditPlugIn.h`. This protocol contains methods for:

- Beginning the edit session.
- Providing the plug-in user interface.
- Using optional callbacks if your plug-in imports new images.

2013-06-05 | Copyright © 2013 Apple Inc. All Rights Reserved.

ApertureEditManager

Specified in the header file `ApertureEditManager.h`. A plug-in uses the methods on this protocol to communicate with Aperture during the edit process. This protocol contains methods for:

- Requesting new, editable versions of the selected images.
- Undoing changes made by the plug-in.
- Adding metadata to images in the Aperture library.
- Importing new images into the Aperture library.
- Ending or canceling the edit session.

PROAPIAccessing

Specified in the header file `PROAPIAccessing.h`. This protocol acts as the broker between Aperture and a plug-in. It allows the plug-in to ask which version of a protocol the host application supports.

Building an Edit Plug-in

The easiest way to build a new Edit plug-in is to create a new project in Xcode using the Aperture Edit Plug-In project template, or to duplicate one of the sample projects. (Both the template and the sample projects are included in the SDK install.) Be sure to customize the class names and to fill in the proper UUIDs to avoid namespace collisions.

Example Edit Plug-ins

An example Edit plug-in is installed in /Developer/Examples/Aperture. This example illustrates API usage as well as several more advanced concepts.

The Info.plist for an Edit Plug-in

An Aperture Edit plug-in is packaged in a CFBundle. Each bundle contains one more plug-ins. A bundle also contains an `Info.plist` file. This file encodes ckey-value pairs that are specific to the ProPlug plug-in architecture. They tell the plug-in manager what protocols and versions the plug-in implements and what host protocols and versions it supports. This file also optionally provides several keys that are specific to Aperture Edit plug-ins.

Here is a commented `Info.plist` template for an Aperture Edit plug-in.

2013-06-05 | Copyright © 2013 Apple Inc. All Rights Reserved.

Listing 1-2 An Edit Plug-in Info.plist

```
<key>ProPlugDictionaryVersion</key>
<string>1.0</string>
<!--Required. Identifies the version of the ProPlug dictionary this plug-in uses.
 The value should be 1.0 for this version of the Aperture SDK.-->

<key>ProPlugDynamicRegistration</key>
<false/>
<!--Required. Tells the plug-in manager whether the plug-in principal class performs
 plug-in registration tasks. If true, the class must implement the
PROPluginRegisterBundle protocol.-->

<key>ProPlugInGroupList</key>
<!--Required, along with all child keys. Identifies this plug-in as an Edit
plug-in.-->
<array>
  <dict>
    <key>groupName</key>
    <string>Edit</string>
    <key>uuid</key>
    <string>616BA321-B4C2-49DF-8FD8-2E3392D2D240</string>
  </dict>
</array>

<key>ProPlugPlugInList</key>
<!--Required. Identifies the plug-in's in this bundle.-->
<array>
  <dict>
  <key>className</key>
  <string>xxxxxxxxxxx</string>
  <!--Required. The name of the class that implements the plug-in protocol.-->

  <key>displayName</key>
  <string>xxxxxxxxxxx</string>
  <!--Required. The name that will appear in menus and on the Edit window.-->
```

```
<key>protocolNames</key>
<array>
<string>ApertureEditPlugIn</string>
</array>
<!--Required. Specifies the plug-in protocol this plug-in implements.This
value should be ApertureEditPlugIn for this version of the Aperture SDK.-->

<key>infoString</key>
<string>Description of your plug-in and what it does</string>
<!-- Optional. This is the descriptive text that displays in the Aperture
Command Customization window.-->

<key>uuid</key>
<string>xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx</string>
<!--Required. Specifies a unique identifier for this plug-in. Use the uuidgen
command-line tool to generate a unique identifier.-->

<key>version</key>
<string>1</string>
</dict>

<key>supportedRAWExtensions</key>
<array>
  <string>cr2</string>
</array>
<!-- Optional. If your plug-in reads the RAW master data from images
but only supports certain raw formats, providing a list of file name extensions
here will disable your plug-in if the user has selected any images that
aren't RAW or are RAW but do not have one of the extensions listed here.
Note that Aperture will still pass images that do not have any extension at
all to your plug-in, regardless of the extensions listed here. -->

<key>supportedEditableExtensions</key>
<array>
```

Copyright © 2013 Apple Inc. All Rights Reserved.

```
    <string>tiff</string>
  </array>
  <!-- If your plug-in only knows how to read or write certain types of files,
  it can request that Aperture write new editable files in those formats.
  However, images may already be editable when the user selects them, meaning
  Aperture will not write a new file for your plug-in. Providing a list
  of supported editable file name extensions here tells Aperture to
  disable your plug-in if the user has selected images that are already editable,
  but whose master file does not have one of the listed extensions. -->
</array>

<key>ProPlugProtocolList</key>
<!--Required, along with all child keys. Specifies the host protocols that this
plug-in supports.-->
<array>
  <dict>
    <key>protocolName</key>
    <string>ApertureEditManager</string>
    <key>versions</key>
    <array>
      <integer>1</integer>
    </array>
  </dict>
</array>
```

Plug-in Locations

Aperture looks for Export and Edit plug-ins in two locations:

```
/Library/Application Support/Aperture/Plug-Ins
~/Library/Application Support/Aperture/Plug-Ins
```

Plug-ins located in the first directory are available to all users of a machine. Plug-ins installed in the second location are available only to the current user. Apple recommends the user folder as the default installation location.

2013-06-05 | Copyright © 2013 Apple Inc. All Rights Reserved.

Before Aperture can display a list of available plug-ins, it recursively scans the plug-in folders for available CFBundles whose primary class conforms to the ApertureExportPlugIn or ApertureEditPlugIn protocol and which also specify this protocol in their `Info.plist` file. Once Aperture has the list of installed plug-ins, it uses other properties in the files, such as display name, group, and description, to create menu items for each plug-in.

2013-06-05 | Copyright © 2013 Apple Inc. All Rights Reserved.

Document Revision History

This table describes the changes to *Aperture SDK Overview*.

Date	Notes
2013-06-05	New document that describes the Aperture SDK released in conjunction with Aperture 3.4.

2013-06-05 | Copyright © 2013 Apple Inc. All Rights Reserved.

Apple Inc.
Copyright © 2013 Apple Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer or device for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-branded products.

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Apple, the Apple logo, Aperture, Mac, OS X, and Xcode are trademarks of Apple Inc., registered in the U.S. and other countries.

APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT, ERROR OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

Some jurisdictions do not allow the exclusion of implied warranties or liability, so the above exclusion may not apply to you.